CSSE 220 Day 27

Data Structures Practice

Checkout *DataStructures* project from SVN

Questions

Common ADTs

- Array List
- Linked List
- Stack
- Queue
- Set
- Map

Implementations for all of these are provided by the Java Collections Framework in the **`java.util`** package.

Array Lists and Linked Lists

Operations Provided	Array List Efficiency	Linked List Efficiency
Random access	O(1)	O(n)
Add/remove item	O(n)	O(1)

Q1,2

Stacks

- A last-in, first-out (LIFO) data structure
- Real-world stacks
  - Plate dispensers in the cafeteria
  - Pancakes!
- Some uses:
  - Tracking paths through a maze
  - Providing "unlimited undo" in an application

Operations Provided	Efficiency
Push item	O(1)
Pop item	O(1)

Implemented by `Stack`, `LinkedList`, and `ArrayDeque` in Java

Q3

Queues

- A first-in, first-out (FIFO) data structure
- Real-world queues
  - Waiting line at the BMV
  - Character on Star Trek TNG
- Some uses:
  - Scheduling access to shared resource (e.g., printer)

Operations Provided	Efficiency
Enqueue item	O(1)
Dequeue item	O(1)

Implemented by **`LinkedList`** and **`ArrayDeque`** in Java

Q4

Sets

- **Unordered** collections **without duplicates**
- Real-world sets
  - Students
  - Collectibles
- Some uses:
  - Quickly checking if an item is in a collection

Operations	HashSet	TreeSet
Add/remove item	O(1)	O(lg n)
Contains?	O(1)	O(lg n)

Can hog space

Sorts items!

Q5

Maps

- Associate keys with values
- Real-world "maps"
  - Dictionary
  - Phone book
- Some uses:
  - Associating student ID with transcript
  - Associating name with high scores

Operations	HashMap	TreeMap
Insert key-value pair	O(1)	O(lg n)
Look up value for key	O(1)	O(lg n)

Can hog space

Sorts items by key!

Q6

Markov Chaining

- Details

Markov Chain Progam

- Input: a text file

```
the skunk jumped over the stump
the stump jumped over the skunk
the skunk said the stump stunk
and the stump said the skunk stunk
```

- Output: a randomly generated list of words that is “like” the original input in a well-defined way

Markov Chain Process

- Gather statistics on word patterns by building an appropriate data structure
- Use the data structure to generate random text that follows the discovered patterns

Markov Example, n = 1

- Input: a text file

the skunk jumped over the stump
the stump jumped over the skunk
the skunk said the stump stunk
and the stump said the skunk stunk

Prefix	Suffixes
NONWORD	the
the	skunk (4), stump (4)
skunk	jumped, said, stunk, the
jumped	over (2)
over	the (2)
stump	jumped, said, stunk, the
said	the (2)
stunk	and, NONWORD
and	the

Markov Example, n = 2

- Input: a text file

the skunk jumped over the stump
the stump jumped over the skunk
the skunk said the stump stunk
and the stump said the skunk stunk

Prefix	Suffixes
NW NW	the
NW the	skunk
the skunk	jumped, said, the, stunk
skunk jumped	over
jumped over	the
over the	stump, skunk
the stump	the, jumped, stunk, said
…	

Output

- n=1:

the skunk the skunk jumped over the skunk stunk

the skunk stunk

- n=2:

the skunk said the stump stunk and the stump jumped over the skunk jumped over the skunk stunk

- Note: it's also possible to hit the max before you hit the last nonword.

Markov Data structures

- For the prefixes?
- For the set of suffixes?
- To relate them?

Prefix	Suffixes
NW NW	the
NW the	skunk
the skunk	jumped, said, the, stunk
skunk jumped	over
jumped over	the
over the	stump, skunk
the stump	the, jumped, stunk, said
…	

Fixed-Length Queue and Markov

- FixedLengthQueue: a specialized data structure, useful for Markov problem
- Check out **`FixedLengthQueue`**
  - Working alone? See your individual repo.
  - Working with a partner? See your Team0x repo.

- Work to implement it in the next 25 minutes or so

Work Time

- Review HW description,
  Work on Markov for rest of class (except when demoing Vector Graphics for me)

Work Time

- Review HW description
- What questions do you have?

- Work on Markov for rest of class…
  - except when demonstrating your Vector Graphics status to me